

# Freakduino Long Range Wireless Board WalkThrough - Basic Usage

[| Print |](#)

Written by Akiba

Sunday, 06 October 2013

This is a walkthrough of the basic setup and usage of the Freakduino long range wireless boards. I originally designed the Freakduino board series and chibiArduino software stack so that it could be a simple way to setup a wireless connection without having to understand complex protocol details. This was a big drawback in many of the more advanced protocol stacks I've worked on where there was complex and detailed knowledge required just to send simple packets. I tried to be minimalistic in the design of the chibiArduino stack so that people who just wanted to do simple wireless communications could do it with many of the protocol details handled in the background.

In this walkthrough, we're going to go through a few examples to set up the boards and write simple code to start transmitting and receiving wireless data.

## Hello World

We're going to start our walkthrough with Hello World. This will take us through some of the basic preparation to make sure our board is functional.

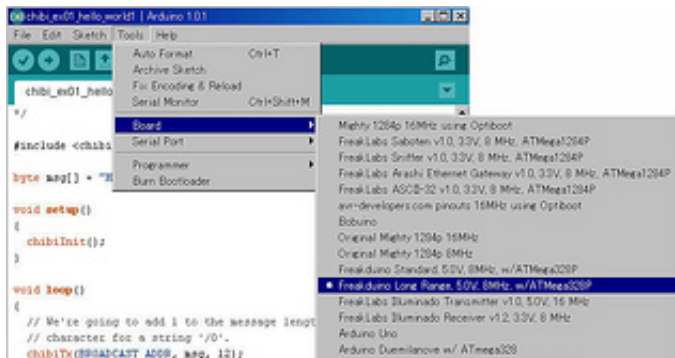
- First, start off by going through the [Freakduino 900 LR User Guide](#) and installing the chibiArduino and hardware support package for your system. The user guide contains instructions to install these for Mac, Linux, and Windows systems.
- Once that's done, we're going to do a simple hello world to make sure the board is functional. Type this into the Arduino IDE:

Code:

```
void setup()
{
  Serial.begin(57600);
}

void loop()
{
  Serial.println("Hello World");
}
```

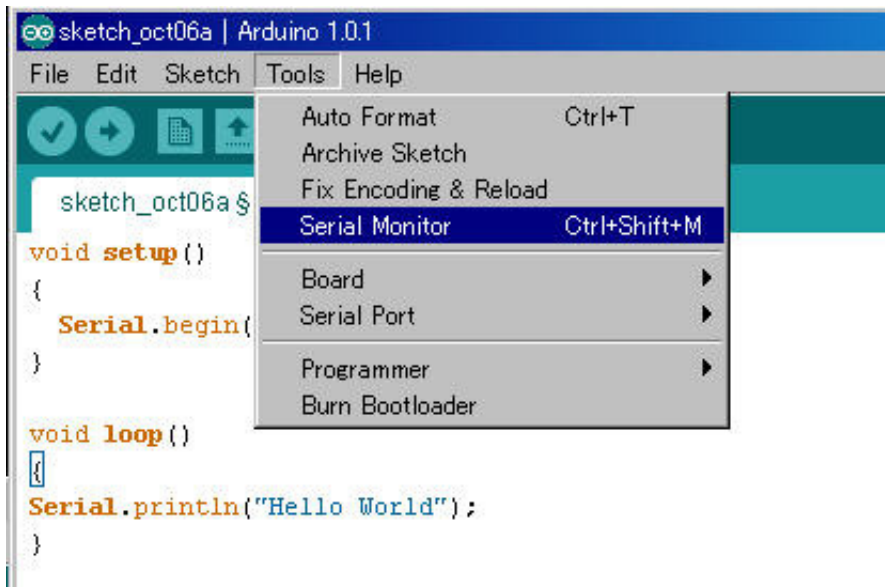
- Choose the board type. If the board support package is installed properly, you should see all the available Freakduino boards listed. Select the Freakduino Long Range, 5.0V, 8MHz, w/ATMega328P. If you don't see any Freakduino boards in the menu, go through the board support package installation again and restart the IDE after you're finished.

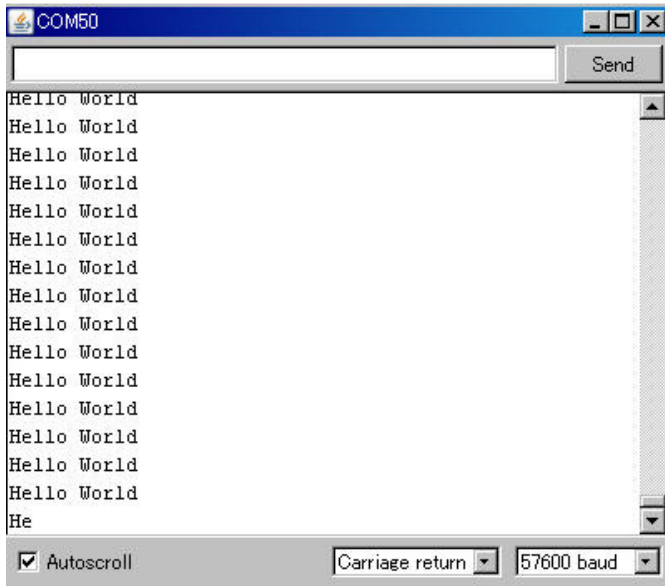


- Upload the code to the board.



- Go to the Arduino IDE and navigate to the Tools/Serial Monitor menu item. Click on the Serial Monitor to open it. If it's not already configured, configure it for a baudrate of 57600 bps. You should see the "Hello World" message infinitely looping. If you can see this, then the board is configured properly and is functional.





---

## Wireless Hello World

Now we're going to add a twist by doing a wireless hello world. This assumes you have two boards that can talk to each other.

- Make sure you have [the most recent chibiArduino library](#) and its installed in the proper directory. The installation instructions should be in the User Manual.
- To use the chibiArduino library, you'll need to include the chibi library in the sketch. Once that's done, the chibiArduino API is available for use.
- In the setup portion of the code, we're going to want to initialize the chibiArduino protocol stack and set any other parameters that are needed. In this case, we're just going to set the addresses for each board. The addresses need to be different. All devices in the same 802.15.4 network need to have different addresses.
- We're going to have two sets of code, one for the transmitter and one for the receiver. The setup code should be the same for both, although the addresses need to be different.

*Transmitter Setup Code:*

```
#include <chibi.h>
#define DEST_ADDR 5 // this will be address of our receiver

void setup()
{
    chibiInit();
    chibiSetShortAddr(3);
}
```

*Receiver Setup Code:*

```
#include <chibi.h>

void setup()
{
    Serial.begin(57600);
}
```

```
chibiInit();
chibiSetShortAddr(5);
}
```

- Next, we're going to write the loop code. This is the meat of the sketch and it will be different for the transmitter and receiver. For the transmitter code, we're going to create a transmit byte array (buffer). This will store the data to be transmitted. The max payload size for the chibiArduino stack is 100 bytes so the buffer will need to be less than or equal to this. I'll be using the strcpy (string copy) function to copy "Hello World" into the data buffer. You'll also see that the dataBuf array needs a typecast to a char type. This is because C/C++ is a strongly typed language so it won't allow copying a char array to an array of a different type. Typecasting will be pretty common when working with the chibiArduino stack since everything is transmitted as a byte array. Finally, to transmit, we'll be using the chibiTx function. This function just takes three arguments: the destination address, the byte array to be transmitted, and the length of the byte array. One thing to note is that when calculating the length to transmit, one byte has been added for the string terminator.

*Transmitter code:*

```
void loop()
{
    byte dataBuf[100];
    strcpy((char *)dataBuf, "Hello World");
    chibiTx(DEST_ADDR, dataBuf, strlen((char *)dataBuf)+1);
    delay(10); // add a bit of a delay so we don't overwhelm the receiver
}
```

- For the receiver code, we're going to poll a flag in the chibiArduino stack to see if any data was received. If data arrives in the radio, it will send an interrupt to the MCU telling it that data has been received. The MCU will then fetch it and set the received flag. When the function call to chibiDataRcvd() is issued, it will then return true. Once we know data has been received, we issue the chibiGetData() function and it will copy the data into the data buffer. Once it's in the data buffer, the sketch can parse the data. We also need to check to see if the length of the data is 0. The stack will set a length of 0 to duplicate packets so we just discard these.

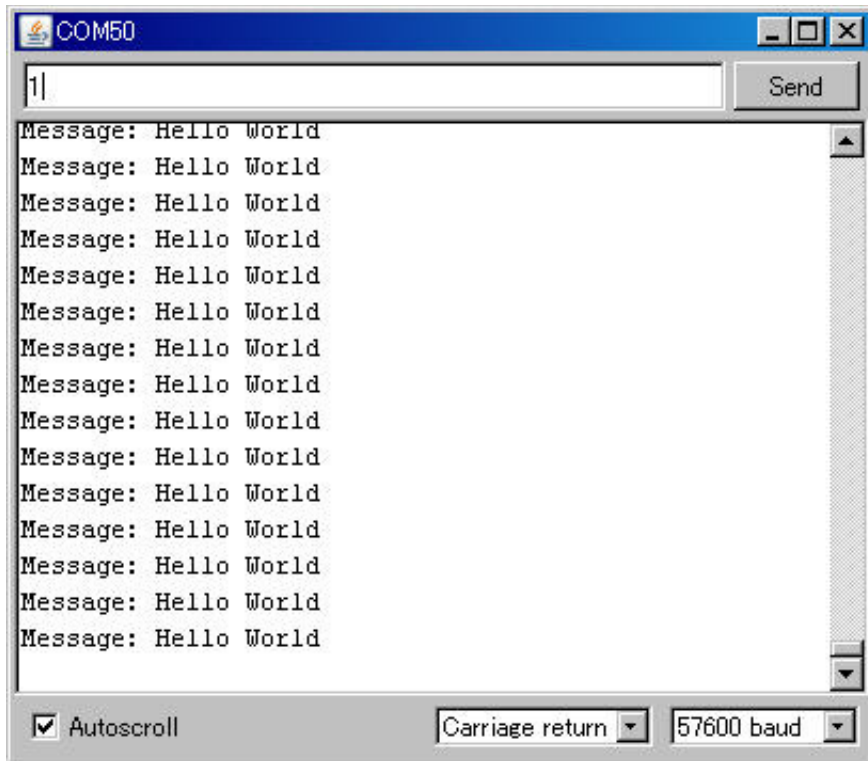
*Receiver code:*

```
void loop()
{
    // Check if any data was received from the radio. If so, then handle it.
    if (chibiDataRcvd() == true)
    {
        byte len, buf[100];

        len = chibiGetData(buf);
        if (len == 0) return; // if no len, its a dupe packet. discard.

        Serial.print("Message: ");
        Serial.println((char *)buf);
    }
}
```

- Now that we have the setup and loop code for the transmitter and receiver, we can put them together and upload them into our boards. Once you do this, open the Serial Monitor on the receiver board. You should see it printing out "Hello World" infinitely.



---

## Freakduino Command Line Control

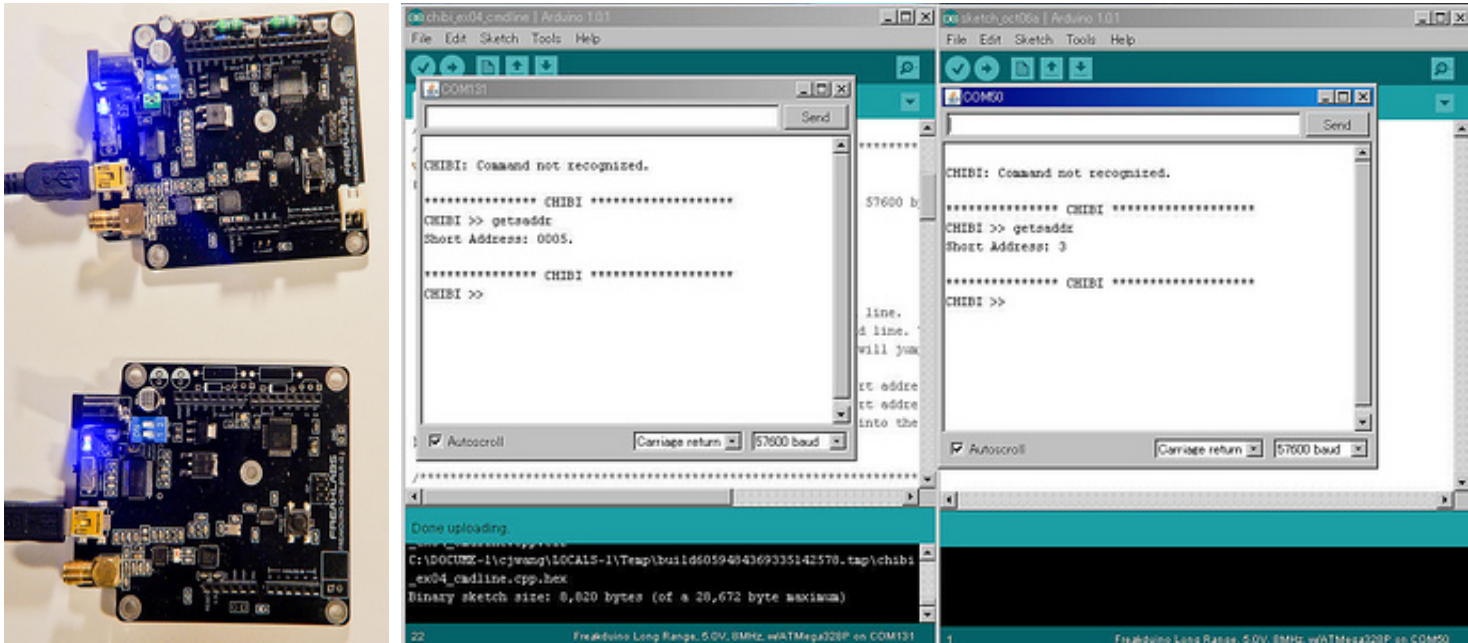
One of my favorite tools to use is [a command line tool I wrote called cmdArduino](#). I actually used it so much during development of the chibiArduino stack that I put it into the chibiArduino stack as a utility. It's proven to be very useful for testing, debugging, and rapid prototyping of code.

A command line allows you to type in and execute commands with arguments. For example, at the command line, you can type in "getsaddr 3" which will retrieve the short address (16-bit address) of the device. You can add custom functions for whatever you want to execute inside the sketch too. I'll have a more in-depth tutorial on using the command line, but for now, we're just going to test it out and execute some pre-written functions.

- For this section, we're going to use Example 4 (chibi\_ex04\_cmdline) from the Examples folder of the chibiArduino library. Go to File/Examples/chibiArduino/Examples and select example 4.
- If you inspect the code, you'll see in the setup() that along with chibiInit(), there's also a chibiCmdInit(57600) function call. This initializes the command line.
- Also in the setup, you'll see the functions "chibiCmdAdd()". The proper format for these functions are: chibiCmdAdd(command name, function name) where command name is the name that gets typed in the command line and function name is the name of the function that gets executed when the command is typed.
- Finally in the loop() section, you'll see a function called chibiCmdPoll(). This is required to poll the command line for keypress events. The rest of the loop code is for receiving and displaying wireless events.
- Load the code into both Freakduino boards. After the code is loaded into the boards, go to Tools/Serial Monitor and open the Serial Monitor.
- Hit <enter> and you should see the command prompt. If you don't see it, wait a bit and hit <enter> again. The FTDI chips take a second or two to initialize.



- Now that the command line is up, you can enter some of the commands. This example sketch supports three commands with the following format:
  - **getsaddr** : This gets the current short address of the device.
  - **setsaddr <addr>** : This sets the short address of the device. The <addr> argument expects a 16-bit hexadecimal so the max address is FFFF. This is the broadcast address so don't use it unless you know what you're doing though.
  - **send <addr> <string>**: This will send a space delimited string to the specified hexadecimal address. For example: "send 3 hello you sexy thang" would send "hello you sexy thang" to the device with an address of 3.
- Let's first start by getting the short address of one of the devices. Type "getsaddr" into the command line and press <enter>. You should see the current address of the device. Now let's set the address. Type "setsaddr 5" into the command line. Then type "getsaddr" again. You should now see the command return the value of 5. When setting the address, the value gets written into the EEPROM which is nonvolatile. That means that even if the device is turned off, it will retain that address. You only need to set the address once and the device should retain the address until it's changed.
- Now set the address of the second device to 3. To do this, type "setsaddr 3". Then type "getsaddr" to verify that the address is 3.
- Both devices now have addresses so we can now send messages to them. First we have to open a serial monitor for each device. This means we need to open two separate instances of the Arduino IDE since one window can only have one serial monitor. Once two instances of the Arduino IDE are open, then connect the USB port for each device. Select the serial port for device 3 in the IDE and open the serial monitor. Then select the serial port for device 5 and open the serial monitor.



- Now we can send messages back and forth. From device 3, type “send 5 hello”. You should see the message “hello” in device 5’s serial monitor. Now do the same from device 5. Type “send 3 hello”. You should now see “hello” in device 3’s monitor. This is a trivial example, but it’s a good example to show how the command line operates and how it can be useful to configure the sketch and also trigger events.

Well, that's the end of this first walkthrough for basic setup and usage of the Freakduino Long Range Wireless board. In the next walkthrough, I'll be going over radio configuration and power management. Hope this tutorial was helpful :)

Hits: 22143

Email This

## Trackback(0)

[TrackBack URI for this entry](#)

## Comments (2)

[Subscribe to this comment's feed](#)

**RADIO NOT INITIALIZED PROPERLY**  
written by Dzaurov Ismail, January 28, 2014

Hi Akiba!  
My name is Ishmael, I work for Emil. Faced with the problem at prededachi words "Hello WORLD". I downloaded and setup void void loop in the transmitter and receiver as shown in the tutorial. But when opening the receiver monitor the message:  
RADIO NOT INITIALIZED properla

I can not understand what it is connected. Separately downloaded code in the receiver and transmitter ports appointed correctly still leaves this message.  
Votes: +0

vote up

vote down

report abuse

...  
written by Akiba, January 28, 2014

Hi Ishmael.  
It's best to discuss support issues via email or via the support forum. If you go to the top of this page, you'll see the forum link. Otherwise, you can also email me directly. I'll need to understand your settings to see what the problem is, but for now, please take it to email or forum.

Thanks.  
Votes: +0


vote up

vote down

report abuse

---

## Write comment

 [Show/Hide comment form](#)

[Close Window](#)